Edge-Based Viscous Method for Mixed-Element Node-Centered Finite-Volume Solvers

Yi Liu[1], Boris Diskin[2], and Hiroaki Nishikawa[3]
National Institute of Aerospace, Hampton, Virginia 23666, USA

William K. Anderson[4], Gabriel C. Nastac[5], Eric J. Nielsen[6], and Li Wang[7]
NASA Langley Research Center, Hampton, Virginia 23681, USA

A novel, efficient, edge-based viscous (EBV) discretization method has been recently developed, implemented in a practical, unstructured-grid, node-centered, finite-volume flow solver, and applied to viscous-kernel computations that include evaluations of meanflow viscous fluxes, turbulence-model and chemistry-model diffusion terms, and the corresponding Jacobian contributions. Initially, the EBV method had been implemented for tetrahedral grids and demonstrated multifold acceleration of all viscous-kernel computations. This paper presents an extension of the EBV method for mixed-element grids. In addition to the primal edges of a given mixed-element grid, virtual edges are introduced to connect cell nodes that are not connected by a primal edge. The EBV method uses an efficient loop over all (primal and virtual) edges and features a compact discretization stencil based on the nearest neighbors. This study verifies the EBV method and assesses its efficiency on mixed-element grids by comparing the EBV solution accuracy and iterative convergence with those of well-established solutions obtained using a cell-based viscous (CBV) discretization method. The EBV solver's memory footprint is optimized and often smaller than the memory footprint of the CBV solver. A multifold speedup is demonstrated for all viscous-kernel computations resulting in significant reduction of the time to solutions for several benchmark mixed-element-grid computations, including simulations of a flow around NASA's juncture-flow model and a hypersonic, chemically reacting flow around a blunt body.

I. Introduction

This paper verifies and assesses a novel edge-based viscous (EBV) method for node-centered, finite-volume, unstructured-grid discretizations of elliptic second-order partial differential operators that represent viscous effects in computational fluid dynamics (CFD) equations. Node-centered edge-based schemes are widely used in unstructured-grid Navier-Stokes solvers for inviscid and viscous fluxes [1-6]. Typical edge-based schemes for viscous fluxes compute solution gradients at edges. The edge gradients are evaluated either by averaging gradients computed at the edge endpoints or by defining an edge-based stencil of grid points [7]. The edge gradient can be augmented with an edge-based derivative [8] or by adding an adjustable term to damp oscillations [9]. Viscous fluxes are computed in an edge loop that avoids duplicate computations inherent in point and cell loops. These edge-based methods benefit from the efficiency of the edge loop but result in large discretization stencils that include neighbors of neighbors. Alternatively, a thin-layer gradient approximation can be used that includes only the edge-based derivatives, resulting in a compact stencil, but may degrade solution accuracy.

The EBV method considered in this paper follows the approach introduced in Ref. [1]. The EBV method essentially mimics the linear finite-element Galerkin approach on tetrahedra, preserves the compact stencil that includes only

[1]Associate Principal Research Engineer, Senior Member AIAA
[2]Senior Research Fellow, Associate Fellow AIAA
[3]Research Fellow, Associate Fellow AIAA
[4]Senior Research Scientist, Computational Aerosciences Branch, Associate Fellow AIAA
[5]Research Scientist, Computational Aerosciences Branch, Member AIAA
[6]Research Scientist, Computational Aerosciences Branch, Associate Fellow AIAA
[7]Research Aerospace Engineer, Computational Aerosciences Branch, Associate Fellow AIAA

immediate neighbors, but uses an edge-based implementation. The EBV method requires any two vertices of the same cell to be connected by an edge. Thus, the EBV implementation is straightforward on tetrahedra. The EBV method groups the operations required for computing viscous fluxes by edge. This recombination removes redundant computations inherent in the cell-based flux evaluation loop and dramatically reduces the time of viscous-kernel computations that include evaluations of meanflow viscous fluxes, turbulence-model and chemistry-model diffusion terms, and the corresponding Jacobian contributions. The EBV method allocates memory to store a few (six or nine) EBV coefficients per edge. The EBV coefficients represent local grid metrics, do not depend on the solution, and can be precomputed for static and rigidly moving grids. The EBV method has demonstrated second-order accuracy for linear and nonlinear diffusion on tetrahedra [10]. With a correction proposed in Ref. [10], the EBV discretization for the Navier-Stokes equations is also second-order accurate on tetrahedral grids.

The EBV method has been implemented in a large-scale Reynolds-averaged Navier-Stokes (RANS) solver, FUN3D [11], which is developed and maintained at the NASA Langley Research Center (LaRC). The baseline FUN3D finite-volume discretization scheme balances fluxes at median-dual control volumes that are centered at grid points. Inviscid fluxes are evaluated at medians of the edges in an efficient edge-based loop. Viscous fluxes use the Green-Gauss theorem to compute gradients at grid cells. For non-tetrahedral cells, the cell gradients at each edge within the cell are augmented with an edge derivative. Because this approach relies on cell-based gradients, the viscous fluxes are computed in a separate cell-based loop. In this paper, the baseline viscous-flux implementation is referred to as the cell-based viscous (CBV) method. The CBV method provides a compact nearest-neighbor stencil. Finite-volume solutions using the CBV method have been extensively verified and validated through formal analysis and applications [12-18].

The EBV method was initially implemented only on tetrahedra [19, 20]. A multifold acceleration of the viscous-kernel computations was observed on tetrahedral grids. For mixed-element grids, a hybrid EBV/CBV method was applied: the EBV method was used on tetrahedra and the CBV method was used on cells of other types. The hybrid EBV/CBV solutions remained accurate, but the EBV efficiency benefits diminished. In this paper, the EBV approach and the EBV efficiency benefits are extended to mixed-element grids.

An EBV implementation is challenging on mixed-element grids because primal edges do not connect some vertices of non-tetrahedral cells. Thus, virtual edges are needed. There are many valid ways to introduce virtual edges. For example, one can assign a virtual edge to any two vertices of a cell that are not connected by a primal edge. This approach is not efficient as it results in many edges on non-tetrahedral grids (13 edges per grid point on hexahedral grids, 10 edges per grid point on prismatic grids). In the current implementation, non-tetrahedral cells of a general mixed-element grid are divided into tetrahedra using an algorithm based on global numbering of grid points [21]. Virtual edges are defined as the edges of the derived tetrahedral grid that are not present in the mixed-element grid. The EBV method is then implemented on the derived tetrahedral grid. The EBV coefficients are stored for primal and virtual edges of the grid. The viscous-kernel computations are conducted in a loop over all edges, primal and virtual, while the inviscid flux computations are conducted in a loop over the primal edges. This approach minimizes the number of edges (seven edges per grid point) and uses the same EBV routines that have already been developed for tetrahedral grids [19, 20].

Although the initial implementation in Ref. [19] is consistent for linear and nonlinear diffusion equations, truncation-error and discretization-error analyses reported in Ref. [10] reveal that the EBV method of Ref. [19] has a slight inconsistency for the Navier-Stokes equations in presence of highly non-smooth viscosity coefficients. The effects of this inconsistency on accuracy of benchmark-flow solutions are indiscernible on coarse and medium practical grids but can be observed on extremely fine grids. In the current study, this inconsistency has been eliminated, and all simulations are performed with the fully consistent second-order accurate EBV discretization.

Additional memory savings and efficiency gains are obtained on mixed-element grids because, for grid points that are surrounded by non-tetrahedral cells, the EBV stencil is significantly smaller than the CBV stencil, resulting in reduction of off-diagonal terms in the Jacobian, reduced memory footprint, and a speedup of the linear solver. In the CBV method, the number of off-diagonal blocks in a Jacobian row corresponding to an interior grid point is determined by the number of individual vertices in the cells that surround this point. On a hexahedral grid, this is on average 26 off-diagonal blocks; on a prismatic grid, this is on average 20 off-diagonal blocks. The Jacobian of the EBV method reported in this article averages 14 off-diagonal blocks per grid point.

The material in the paper is presented in the following order. Section II outlines the baseline finite-volume discretization methods and iterative solvers. Section III provides an overview of the previously documented implementation and performance of the EBV method on tetrahedral grids. Section IV details the current EBV implementation on mixed-element grids. Section V verifies the EBV mixed-element implementation by comparing the EBV and CBV RANS solutions for a benchmark turbulent flow on a family of prismatic-hexahedral grids and analyzes the EBV efficiency benefits. Section VI presents the EBV solutions for a subsonic flow around NASA's juncture-flow model and for a high-enthalpy, chemically reacting, hypersonic flow around a hemisphere-cylinder configuration. Section VII summarizes the observed results.

II. Baseline Discretization and Iteration Methods

This section presents components of the baseline finite-volume discretization. Nonlinear iterative solvers that are used in this study are also outlined.

A. Baseline Discretization Scheme

The baseline finite-volume solver is used for equations discretized on unstructured mixed-element grids that may contain tetrahedra, pyramids, prisms, and hexahedra. The residuals are evaluated on a set of median-dual control volumes centered at grid points. Edge-based inviscid fluxes are computed at primal edge medians using an approximate Riemann solver. In the current study, Roe's flux-difference splitting [22] is used. For second-order accuracy, density, pressure, and velocity are reconstructed by a UMUSCL (Unstructured Monotonic Upstream-centered Scheme for Conservation Laws) scheme [23, 24]. The negative variant (SA-neg) [25] of the Spalart-Allmaras turbulence model [26] is used in this study. The spatial discretization of the SA-neg turbulence model uses a first-order accurate convection scheme. For the discretization of viscous fluxes, the Green-Gauss theorem is used to compute cell-based gradients. On tetrahedral grids, this CBV approach is equivalent to a Galerkin approximation [1]. For non-tetrahedral grids, cell-based Green-Gauss gradients are combined with edge-based gradients [9, 27, 28] to improve stability of viscous operators and prevent odd-even decoupling. In this paper, the edge-normal augmentation [9, 28] is used. The diffusion terms in the turbulence and chemistry models are handled similarly. The vorticity-based source term for the turbulence model is computed using velocity gradients evaluated by the Green-Gauss method on dual control volumes. Boundary conditions are discussed in Ref. [29], including subsonic pressure-based outflow, farfield based on the Roe's Riemann solver, symmetry, and viscous-wall boundary conditions used in this study.

B. Baseline Iterative Solver

The baseline nonlinear iterations use a defect-correction method with nonlinear residuals representing the target operator and an approximate Jacobian representing the driver operator. For RANS solutions, the meanflow and SA-neg Jacobians are decoupled. For chemically reacting flows, the meanflow and chemistry-model Jacobians are fully coupled. The approximate Jacobian for the meanflow equations is formed using the linearization of the first-order flux-vector splitting inviscid fluxes [30] and the second-order viscous fluxes. The approximate Jacobians for the SA-neg and chemistry-model equations include the contributions from the advection, diffusion, and source terms. The exact linearization is used for the advection and diffusion terms. The pseudotime term is controlled by a Courant-Friedrichs-Lewy (CFL) number that can be predefined or ramped linearly within a specified number of nonlinear iterations. Nonlinear iterations can reuse the Jacobian computed at a previous iteration; Jacobian updates are scheduled according to the convergence of the nonlinear residual. Every tenth nonlinear iteration always performs the Jacobian update; other nonlinear iterations may skip the update if the residual reduction in the previous nonlinear iteration exceeded a specified target. This option is referred to as "smart" Jacobian update. The smart Jacobian update is performed separately for the meanflow and SA-neg Jacobians. At each nonlinear iteration, the linear system is solved using a specified number of point-implicit Gauss-Seidel (GS) sweeps with multicolor ordering. The resulting linear solution is added to the nonlinear solution.

C. HANIM Iterative Solver

Hierarchical adaptive nonlinear iteration method (HANIM) is a strong nonlinear solver that is based on a hierarchy of modules including a preconditioner, a matrix-free linear solver, realizability check, nonlinear control, and CFL adaption modules [31-36]. HANIM enhances the baseline iterative solver with a mechanism for an automatic adaption of the pseudotime step to increase convergence rate and overcome instabilities occurring in transient solutions. The HANIM preconditioner is similar to the baseline defect-correction method. The matrix-free linear solver [37-39] uses Fréchet derivatives and a generalized conjugate residual (GCR) [37] method from the family of Krylov methods. HANIM prescribes the residual reduction targets for the preconditioner, the GCR solver, and nonlinear solution updates and specifies the maximum number of linear iterations allowed in the preconditioner and the maximum number of search directions to be used in the GCR solver. HANIM increases the CFL number if all the HANIM modules have reported success. On the other hand, if any of the modules fails, HANIM discards the correction and aggressively reduces the CFL number. HANIM performs Jacobian updates at the beginning of each nonlinear iteration.

III. Edge-Based Viscous (EBV) Method for Tetrahedral Grids

The EBV method for tetrahedra was derived, analyzed, and applied for diffusion and RANS equations in Refs. [10, 19, 20]. The EBV method precomputes and stores EBV coefficients at each edge. As was rigorously proved in Ref. [20], the 3×3 matrix of the EBV coefficients required for the Navier-Stokes equations is symmetric for interior edges that are fully surrounded by tetrahedra. Six coefficients fully define a symmetric 3×3 matrix. However, symmetry of the EBV matrix is not established for the boundary edges, where three additional coefficients were used. The EBV implementation reported in Ref. [20] allocated memory for nine EBV coefficients at each edge.

The EBV and CBV solutions computed by the baseline iterative solver were compared for a benchmark flow on a family of tetrahedral grids. The residual reduction per iteration and final solutions computed with the EBV and CBV methods were almost identical. For the meanflow viscous fluxes and Jacobian evaluation, the EBV speedup of at least 59% (a speedup factor of 2.5) was observed. The EBV speedup is defined as the difference between the CBV time and the EBV time divided by the CBV time and multiplied by 100%. The EBV speedup factor is defined as the CBV time divided by the EBV time. The implementation of the baseline CBV method for the meanflow fluxes and Jacobian is highly optimized, which makes this significant EBV efficiency gain more impressive. For evaluations of the diffusion term of the SA-neg turbulence model and the corresponding Jacobian contributions, the EBV speedup was much higher, at least 94% (a speedup factor of 17). See Ref. [20] for more details.

The truncation error analysis reported in Ref. [10] revealed that the EBV discretization described in Ref. [19] is missing some terms resulting in a loss of consistency. The loss of consistency was also observed in discretization error analysis for manufactured solutions on extremely fine grids. The missing terms affect momentum and energy conservation equations and include products of velocity gradients and viscosity gradients. In the EBV implementation used in this paper, appropriate source terms are added to the affected equations. The truncation and discretization analyses of the corrected EBV formulation confirm the second-order accuracy for the same manufactured solutions [10]. The EBV correction terms have been added to the residuals of the momentum and energy conservation equations but currently are not incorporated into the approximate Jacobian used in iterative solvers. Note that this correction has a negligible effect on EBV solutions for the benchmark flows reported in Refs. [19, 20], partially because the grids used in those studies are not sufficiently fine to discern the loss of consistency.

In Ref. [20], a hybrid EBV/CBV method was used to compute solutions on mixed-element grids. The EBV method was applied to edges that belong to tetrahedra and the CBV method was applied on non-tetrahedral cells. The hybrid EBV/CBV solutions proved to be accurate and matched the CBV solutions and convergence history almost perfectly. However, the hybrid method produced no efficiency gains because the domain decomposition method used in that study is solely based on equidistribution of point-based operations and results in some partitions that contain only non-tetrahedral cells.

In this paper, an alternative and far more efficient EBV approach to discretization of viscous terms on mixed-element grids is proposed. The non-tetrahedral cells of a mixed-element grid are divided into tetrahedra using the algorithm proposed in Ref. [21]. The derived tetrahedral grid has the same grid points and contains all primal edges of the original mixed-element grid. The residuals of the conservation law equations are defined at grid points. The inviscid fluxes are discretized on the original mixed-element grid. The residual contributions from the inviscid fluxes and the corresponding Jacobian terms are computed in a loop over primal edges. The EBV terms are computed on the derived tetrahedral grid. The EBV residual contributions and the Jacobian terms are computed in a separate loop over primal and virtual edges.

IV. Implementation of EBV Method on Mixed-Element Grids

On a mixed-element grid, the EBV method is formulated on a derived tetrahedral grid that has the same grid points as the target mixed-element grid. For this reason, the formulation of the EBV method for mixed-element grids is essentially the same as the formulation for tetrahedral grids reported in Refs. [19, 20] with correction source terms added to the momentum and energy conservation equations. However, the EBV implementation on mixed-element grids is significantly different from the implementation on tetrahedra. This section describes the implementation details of the EBV method for mixed-element grids.

A. Precomputing EBV coefficients

The EBV preprocessing is conducted following the standard startup procedure of loading and partitioning a mixed-element grid and creating all necessary data structures for grid points, edges, faces, and cells. Within each local partition, non-tetrahedral cells are divided into tetrahedra. The division based on the global grid-point numbering [21] is unique, ensures consistency of the derived tetrahedral grid across the partitions, and adds virtual edges to the primal edges of the mixed-element grid. An EBV edge pointer array of the dimensions (2, number of EBV edges) is created for edges of the derived tetrahedral grid; the EBV edges include all primal and virtual edges. Thus, there are two sets of edges: the set of primal edges of the mixed-element grid is used to compute inviscid fluxes, and the set of EBV edges is used to compute viscous fluxes.

An auxiliary edge-based array is created for pointers to the compressed-row data structure for the Jacobian off-diagonal terms. This auxiliary array has dimensions of (2, number of EBV edges) and facilitates access to the off-diagonal terms of the Jacobian at each edge. Another auxiliary array is created to mark all boundary grid points. This marking distinguishes the boundary EBV edges from the interior EBV edges. The EBV coefficients are precomputed in a loop over cells of the derived tetrahedral grid. See Ref. [20] for more details. The EBV data structures include two two-dimensional four-byte integer arrays for local cell-to-node and cell-to-edge data structures of the derived tetrahedral grid; two four-byte integer edge-pointer (2, number of EBV edges) arrays to point to the edge endpoints and to the Jacobian off-diagonal terms; a one-dimensional integer (can be logical) array for boundary-point indicators; two two-dimensional floating-point edge-based arrays, one array to store the six EBV coefficients per edge for a symmetric EBV matrix and another array to store additional three EBV coefficients per edge for a non-symmetric EBV matrix. For static and rigidly moving grids, the EBV cell-to-node and cell-to-edge data structures are needed only for the preprocessing stage and can be deallocated after the EBV coefficients have been precomputed. All other EBV data structures are used in the solver execution and carried through the entire simulations.

B. EBV method implementation and execution

The first step in the EBV implementation is evaluation of the velocity and viscosity gradients at grid nodes required for computing the correction source terms. The gradients of velocity components computed by an unweighted least-squares method are already available at grid points. These velocity gradients are later used for inviscid-flux UMUSCL reconstruction. The viscosity gradients are also computed by the unweighted least-squares method in an edge-based loop. Currently, the laminar and eddy viscosity coefficients are recomputed from the primitive variables (density, velocity, and pressure) at each edge endpoints. Gradients of the laminar and eddy viscosity are computed and stored at grid points separately. This is a suboptimal implementation that can be improved by locally precomputing the combined viscosity coefficients in a loop over grid points and computing a single gradient for the combined viscosity.

The EBV method for evaluation of the meanflow viscous fluxes is implemented in a separate edge loop, in which each edge provides contributions to the meanflow residuals at the edge midpoints. The edge residual contributions are products of EBV coefficients at the edge representing grid metrics, solution edge differences, and viscosity edge averages [19, 20]. The EBV edge loop is preceded by a local loop over grid points, in which temperature and viscosity coefficients are precomputed and stored, and the correction terms are computed and added to the residuals. The point values of temperature are used to compute the temperature edge difference that is needed for the evaluation of the heat flux. The point values of viscosity are used to compute the edge averages. The edge averages of velocity are computed separately. In Ref. [20], it was shown that the matrix of EBV coefficients is symmetric with respect to the edge endpoints for any interior edge that is fully surrounded by tetrahedra. Only six EBV coefficients are needed for edges with symmetric EBV matrices. Nine EBV coefficients are needed for an edge where symmetry of the EBV matrix is not established. Thus, in the current implementation, all edges are assigned six EBV coefficients, and all boundary edges, i.e., edges with both end-points on a boundary, are assigned three additional EBV coefficients.

The EBV method for the SA-neg diffusion term and its Jacobian is implemented in a separate edge loop. Only one EBV coefficient is needed for evaluation of the SA-neg diffusion term and passed to the corresponding subroutine. This EBV coefficient is the trace of the EBV matrix. No additional storage is needed for the EBV implementation of the SA-neg diffusion term. For multi-species, multi-temperature chemistry models, the EBV diffusion terms of species and energy conservation equations are implemented inside the edge loop that computes viscous terms for the meanflow equations. At each edge, the diffusion terms of the chemistry equations use the same EBV coefficient as the SA-neg diffusion term, so no additional memory is needed for the EBV implementation of chemistry models.

The array to store the off-diagonal terms of the EBV Jacobian matrix is implemented to include only terms that correspond to grid points connected by the EBV edges. For mixed-element grids, this EBV implementation results in a significant reduction in the size of the off-diagonal Jacobian array, a smaller memory footprint for the EBV method, and faster matrix-vector multiplication occurring within the linear solver. For tetrahedral grids, the off-diagonal Jacobian array is the same for the CBV and EBV methods. The EBV memory savings and linear-solver efficiency gains are diminished on grids dominated by tetrahedra. The EBV efficiency gains related to viscous-kernel computations are realized on all grids.

Some boundary conditions implemented in the baseline solver use cell-based gradient evaluation. In presence of such boundary conditions, the rows of the EBV off-diagonal Jacobian matrix corresponding to a boundary grid point are extended to include off-diagonal terms for all vertices of the cells that share the boundary grid point and have a face at a boundary surface.

V. EBV Solutions on Mixed-Element Grids

This section compares the CBV and EBV solutions on a family of prismatic-hexahedral grids. Fully converged RANS solutions are computed with the SA-neg turbulence model for a benchmark flow that is widely used for verification of RANS solvers. Viscous-kernel computations associated with the CBV and EBV methods are profiled. Performance of the baseline and HANIM solvers is also assessed.

A. Benchmark flow conditions and prismatic-hexahedral grid family

The three-dimensional benchmark flow considered in this section is a subsonic separated flow around a hemisphere-cylinder configuration [40]. This benchmark flow is described at the NASA Turbulence Modeling Resource (TMR) website[§] and used in Refs. [19, 20] to demonstrate the EBV method on tetrahedral grids. The cylinder and hemisphere have diameters of unity. The combined length of the configuration is 10. The apex of the hemisphere is located at the origin of the coordinate system. The cylinder axis is aligned with the x-axis. The outflow conditions are assigned at a plane that is orthogonal to the x-axis and contains the cylinder base located at $x = 10$. The symmetry condition is assigned at the vertical plane corresponding to $y = 0$. The farfield boundary is a quadrant of a sphere $((x-10)^2 + y^2 + z^2 = r^2,\ (10-r) \le x \le 10,\ 0 \le y \le r,\ -r \le z \le r)$ with the radius $r = 100$. The flow corresponds to the reference (freestream) Mach number of 0.6, the Reynolds number of 3.5×10^5 based on the unit length, and an angle of attack of 19°.

A family of four nested, prismatic-hexahedral grids, denoted as PH0, PH1, PH2, and PH3, has been generated using the grid generation and coarsening programs available at the TMR website[†]. The PH0 grid with about 71 million grid points and about 1.2 million quadrilateral and 24.6 thousand triangular boundary faces has been generated first. The PH1, PH2 and PH3 grids are derived from the PH0 grid using the grid-coarsening program. Figure 1 shows the volume and surface meshes corresponding to the PH3 grid; red color indicates the cylinder surface, blue color shows the symmetry boundary, green color shows the outflow boundary, and orange color marks the farfield boundary. Table 1 provides the grid statistics and the number of Central Processing Unit (CPU) cores used for each grid. The PH1, PH2, and PH3 mixed-element grids use the same grid points as the corresponding tetrahedral grids in Refs. [19, 20]. On these grids, the number of virtual edges is greater than the number of the primal edges by about 20%. The grids PH1, PH2 and PH3 are partitioned so that each individual partition has approximately 30,000 grid points; each partition of the PH0 grid has about 180,000 grid points. The solutions are computed on the NASA LaRC K4 cluster using Intel Gold 6148 Skylake compute nodes. Each node has 40 2.4GHz CPU cores. Intel® Fortran 2019 compiler has been used with the optimization level "–O2", which is standard for FUN3D production runs.

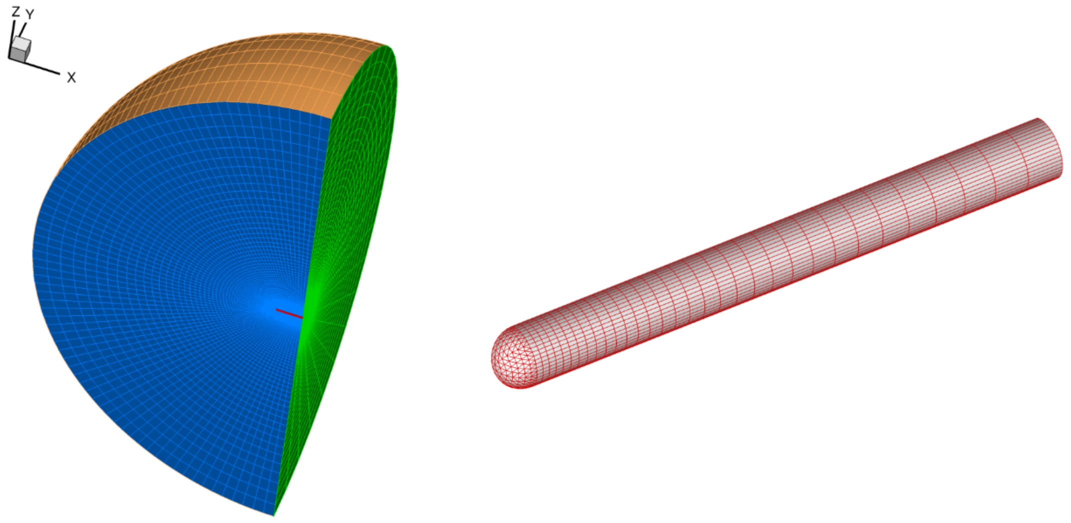


Figure 1. Volume and surface mesh for benchmark flow around hemisphere cylinder.

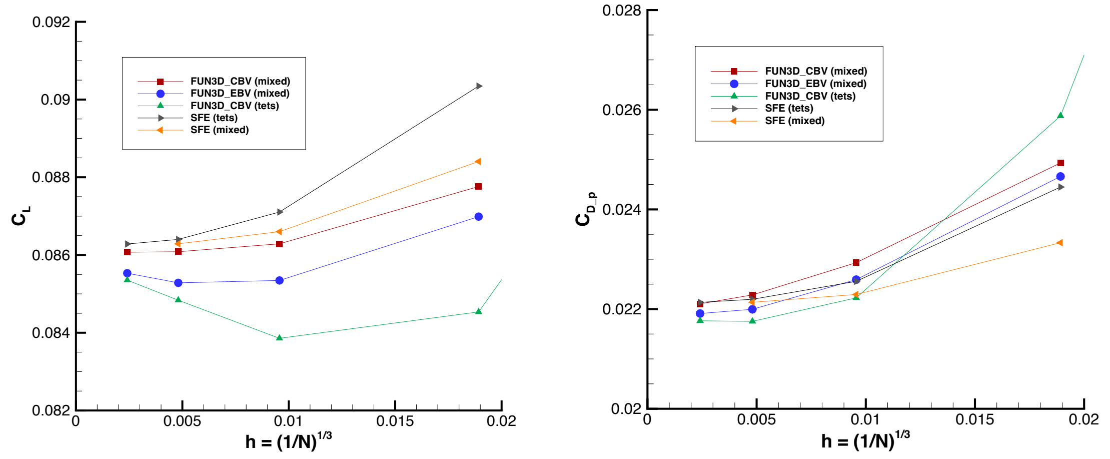
[§]https://turbmodels.larc.nasa.gov/hc3dnumericspart2_val.html. Accessed March 21, 2022.
[†]https://turbmodels.larc.nasa.gov/hc3dnumericspart2_grids.html. Accessed March 21, 2022.

Table 1. Family of prismatic-hexahedral grids for hemisphere cylinder.

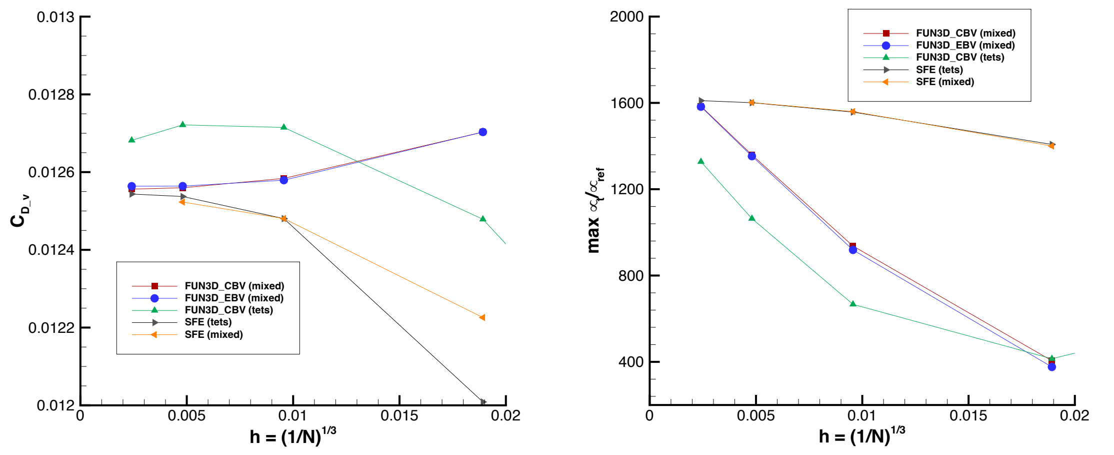
Grid	Points	Hexahedra	Prisms	Primal Edges	Virtual Edges	CPU cores
PH0	71,368,353	62,914,560	15,728,640	221,384,352	275,832,832	400
PH1	8,995,153	7,864,320	1,966,080	27,821,392	34,551,808	320
PH2	1,143,081	983,040	245,760	3,514,920	4,337,152	40
PH3	147,637	122,880	30,720	448,756	546,688	5

B. Verification of EBV method

To verify the EBV implementation, EBV solutions have been computed on the family of prismatic-hexahedral grids and compared with established reference solutions from the TMR website. The reference solutions [13] are computed on mixed-element and tetrahedral grids with the finite-volume CBV method (FUN3D_CBV) and with a finite-element method corresponding to a second-order accurate stabilized Petrov-Galerkin discretization (SFE). Figure 2 shows the grid convergence of the lift, pressure-drag, and viscous drag coefficients, and the maximum eddy viscosity. The eddy viscosity is nondimensionalized by the reference dynamic viscosity corresponding to the freestream conditions. The EBV solutions on each grid are well within the range of the reference solutions computed with the same degrees of freedom. On the finer grids, the EBV lift and pressure drag coefficients are aligned with the corresponding coefficients computed by the CBV method on tetrahedral grids. The EBV viscous drag coefficient and the maximum eddy viscosity are especially close to those computed by the CBV method on mixed-element grids.



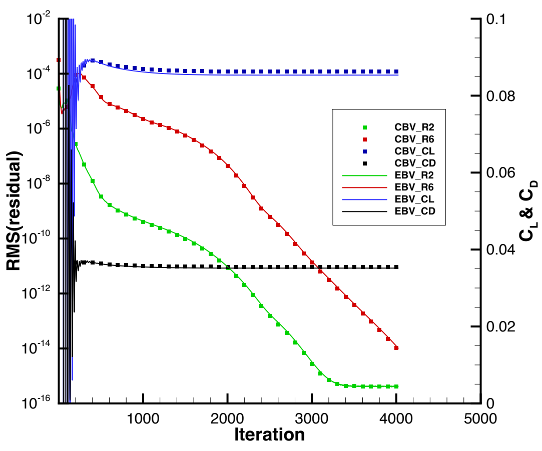
a. Lift coefficient b. Pressure drag coefficient



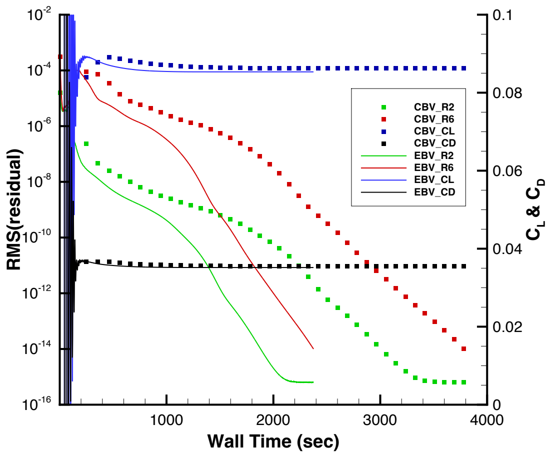
c. Viscous drag coefficient d. Maximum eddy viscosity

Figure 2. Grid convergence of aerodynamic coefficients.

The baseline iterations are compared for the EBV and CBV methods. The iteration stopping criterion is set as 10^{-14} for the root-mean-square (rms) norm of residuals. The maximum CFL number is set to 100. The decoupled linear system is solved using 30 multicolor sweeps for the meanflow and 15 multicolor sweeps for the SA-neg equation. The UMUSCL parameter is set to $\kappa = 0.5$. The "smart" Jacobian update is used. Figures 3 and 4 show iterative convergence on the PH2 and PH1 grids, respectively. Subscripts $R2$ and $R6$ denote the rms norms of the x-momentum and SA-neg residuals, respectively; subscripts CL and CD and axis labels C_L and C_D denote the lift and drag coefficients. The EBV and CBV solutions show similar convergence per iteration and a close agreement between the converged aerodynamic coefficients. The EBV and CBV residual convergence plots shown in Figs. 3a and 4a versus iterations are hardly distinguishable. Although, the rms norms of other RANS residuals are not shown, they all exhibit similar trends. Such a close similarity between the CBV and EBV iterative convergence plots is somewhat surprising as the CBV and EBV discretizations of viscous terms on non-tetrahedral grids are expected to be significantly different. Apparently, the CBV and EBV discretizations are close not only on tetrahedral grids, but on orthogonal non-tetrahedral grids too. The iterative convergence history versus wall time in seconds is shown in Figs. 3b and 4b. The EBV method takes less time per iteration than the CBV method and significantly reduces the time to converge residuals and aerodynamic forces. Comparing with the corresponding tetrahedral-grid solutions reported in Ref. [20], both the CBV and EBV methods use fewer iterations and converge faster on the prismatic-hexahedral grids.

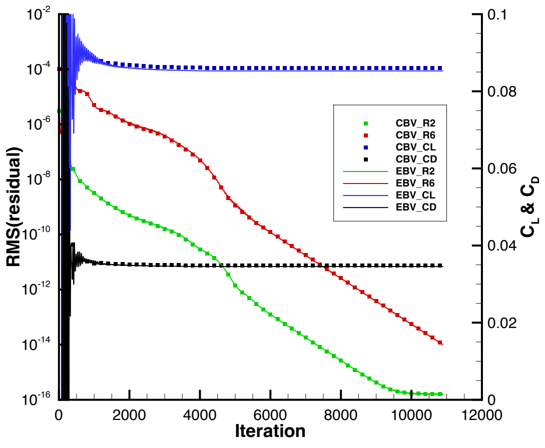


a. Nonlinear iterations

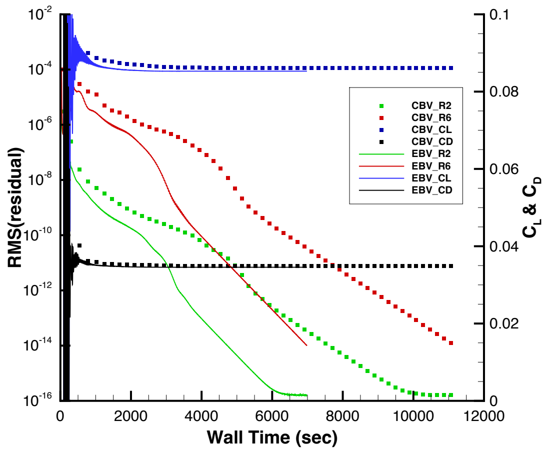


b. Wall time

Figure 3. Baseline iterations on PH2 grid.



a. Nonlinear iteration



b. Wall time

Figure 4. Baseline iterations on PH1 grid.

Viscous-kernel computations and linear solver

In distinction from the EBV performance on tetrahedral mesh [20], the EBV speedup on mixed-element grids is a composition of the speedup of the viscous-kernel computations and the speedup of the linear solver. Tables 2, 3, and 4 compare the wall time used by the CBV and EBV methods for viscous-kernel and linear-solver computations on the PH1, PH2, and PH3 grids. The data corresponding to the PH0 grid is not shown because the PH0 grid partitions are not consistent with the partitions of the coarser grids. The timing shown in Table 1 has been averaged over all iterations and recorded separately for the meanflow and SA-neg equations. The EBV method shows at least 76% speedup (a speedup factor of 4.2) in computing the viscous-flux contributions to the meanflow residuals, 90% speedup (a speedup factor of 9.7) in computing the diffusion contributions to the SA-neg residual, about 70% speedup (a speedup factor of 3.25) in computing the viscous-flux contributions to the meanflow Jacobian, 95% speedup (a speedup factor of 20) in computing the diffusion contributions to the SA-neg Jacobian, about 40% speedup in the linear solver for the meanflow equations, and between 21% and 27% speedup in the SA-neg linear solver.

Table 2. Time and speedup for meanflow viscous flux and SA-neg diffusion evaluation.

Grid	Meanflow viscous fluxes			SA-neg diffusion		
	CBV (sec)	EBV (sec)	EBV speedup	CBV (sec)	EBV (sec)	EBV speedup
PH1	0.0498	0.0108	78%	0.0456	0.0041	91%
PH2	0.0423	0.0101	76%	0.0386	0.0037	91%
PH3	0.0275	0.0053	81%	0.0251	0.0026	90%

Table 3. Time and speedup for Jacobian evaluation.

Grid	Meanflow viscous fluxes			SA-neg diffusion		
	CBV (sec)	EBV (sec)	EBV speedup	CBV (sec)	EBV (sec)	EBV speedup
PH1	0.2359	0.0693	71%	0.1238	0.0068	95%
PH2	0.2274	0.0674	70%	0.1136	0.0063	95%
PH3	0.1563	0.0484	69%	0.0794	0.0042	95%

Table 4. Time and speedup for linear solver.

Grid	Meanflow			SA-neg		
	CBV (sec)	EBV (sec)	EBV speedup	CBV (sec)	EBV (sec)	EBV speedup
PH1	0.6981	0.4244	39%	0.0223	0.0166	26%
PH2	0.6460	0.3932	39%	0.0196	0.0143	27%
PH3	0.3401	0.1957	43%	0.0105	0.0084	20%

In comparison with the results on tetrahedral grids reported in Ref. [20], the EBV speedup for viscous-kernel computations on mixed-element grids is significantly higher for the meanflow computations and a little lower for the SA-neg computations. The CBV time on prismatic-hexahedral grids has increased by a factor of two comparing to the CBV time on tetrahedral grids with the same grid points. Some increase is expected as the number of operations performed in each non-tetrahedral cell is increased dramatically overweighting the reduction in the number of cells. Somewhat surprisingly, the timing for the EBV computations on the two finer mixed-element grids has also increased in comparison with the timing on the corresponding tetrahedral grids, although not significantly. This minor EBV slowdown on mixed-element grids is attributed to suboptimal grid partitioning that does not account for EBV virtual edges. The EBV speedup in linear solver occurs only on non-tetrahedral grids.

There is a factor of seven difference between the time of the EBV viscous flux evaluation and the EBV Jacobian evaluation. This difference is even larger than the factor-six difference observed on tetrahedral grids [20]. This increase is explained by a new separate-loop implementation of the EBV Jacobian. In Ref. [20], the EBV Jacobian was implemented in the same edge loop with the Jacobian for the inviscid fluxes. The corresponding difference between the SA-neg diffusion evaluation and the corresponding Jacobian is smaller (less than factor of two), but still significant. A similar timing is theoretically expected for residual and Jacobian evaluations. The increased time for the meanflow Jacobian evaluation is in part explained by the need to convert from primitive to conserved variables. A suboptimal partitioning may be another factor leading to the increased Jacobian time. In single-processor computations, the time difference between the SA-neg diffusion and Jacobian evaluations was less than 2%.

C. Baseline iterations

The EBV speedup of an individual nonlinear iteration depends on the fraction of time that is spent on viscous-kernel computations. This fraction is significantly higher when the Jacobian is updated. Comparison of CBV and EBV timing for a single nonlinear iteration is shown in Table 5. The timing for such computations has been averaged over five nonlinear iterations. The overall speedup for a nonlinear iteration that updates Jacobian varies from 44% to 47% (a speedup factor of 1.8-1.9). The speedup for a "lean" nonlinear iteration that does not update Jacobian is between 35% and 38% (a speedup factor of 1.5-1.6).

Table 5. Time and speedup for nonlinear iteration on prismatic-hexahedral grid.

Grid	Without Jacobian update			With Jacobian update		
	CBV (sec)	EBV (sec)	EBV speedup	CBV (sec)	EBV (sec)	EBV speedup
PH1	0.9438	0.6087	36%	1.4138	0.7938	44%
PH2	0.8529	0.5558	35%	1.3470	0.7475	45%
PH3	0.4747	0.2946	38%	0.7968	0.4200	47%

The overall performance of the baseline solver using CBV and EBV methods is quantitatively assessed in Tables 6 and 7. These are the same iterations as in the verification study described in Section VI.B. Table 6 shows the fraction of time spent in each type of viscous-kernel and linear-solver computations. The fraction is computed from the wall time that is required to reach the converged solution from the freestream setup. In these tests, the fraction of all viscous-kernel computations is relatively small in comparison with the fraction of the linear solver; the latter takes about two thirds of the entire computing time. Because the baseline nonlinear iterations use smart Jacobian update, there are fewer Jacobian updates than the number of iterations. The EBV method dramatically reduces the fractions of all viscous-kernel computations, especially, on finer grids. On the PH1 grid, the reduction factor ranges from 2.1 (meanflow Jacobian) to 14 (SA-neg Jacobian). The fraction of the linear solver time remains about the same, indicating that the most of the overall EBV speedup is due to speedup of the linear solver.

Table 6. Fractions of viscous-kernel and linear-solver computations on prismatic-hexahedral grids.

Grids	Viscous method	Iterations	Meanflow				SA-neg			
			Viscous flux fraction	Jacobian		Linear solver fraction	Diffusion fraction	Jacobian		Linear solver fraction
				Updates	Fraction			Updates	Fraction	
PH1	CBV	10,908	4.9%	1,512	3.2%	68.3%	4.5%	2,499	2.8%	2.2%
	EBV	10,897	1.7%	1,510	1.5%	66.5%	0.6%	2,501	0.2%	2.6%
PH2	CBV	4,005	4.5%	578	3.5%	68.4%	4.1%	1,033	3.1%	2.1%
	EBV	4,026	1.7%	580	1.6%	66.7%	0.6%	1,034	0.3%	2.4%
PH3	CBV	2,199	5.2%	334	4.5%	63.8%	4.7%	543	3.7%	2.0%
	EBV	2,192	1.7%	334	2.3%	61.7%	0.8%	545	0.3%	2.7%

Table 7 shows the wall time to solution for the baseline solver with the CBV and EBV methods. The EBV speedup above 37% (a speedup factor of 1.6) is observed on all grids. This EBV speedup on prismatic-hexahedral grids is significant because the baseline CBV solver on such grids is fast. For example, the PH2 CBV solution converged almost eight times faster than the CBV solution on the corresponding tetrahedral grid [20]. Although not shown, the EBV speedup of the wall time to solution for the baseline solver on the PH0 grid is 42.4%.

Table 7. Time and speedup for baseline solver on prismatic-hexahedral grids.

Grid	CBV (sec)	EBV (sec)	EBV speedup
PH1	11,146	6,959	37.6%
PH2	3,782	2,372	37.3%
PH3	1,171	696	40.6%

Table 8 shows the memory requirements for the CBV and EBV solvers. The memory usage is reported by the portable batch system (PBS), the job scheduling software that was employed to schedule the computations. The current code carries all CBV data structures along with the EBV solver penalizing the EBV memory consumptions. In spite of this penalty, the EBV solver requires less memory than the CBV solver, saving up to 4.4% of the memory resources. The memory saving percentage increases on finer grid. Although not shown, the EBV method saves 8.9% memory (as reported by PBS) on the PH0 grid. These data indicate that on prismatic and hexahedral cells, the memory reduction due to fewer off-diagonal terms in the EBV Jacobian is more than sufficient to compensate the additional memory allocation for virtual edges and EBV coefficients. The memory advantage of the EBV method over the CBV method is expected to diminish and reverse on grids dominated by tetrahedra.

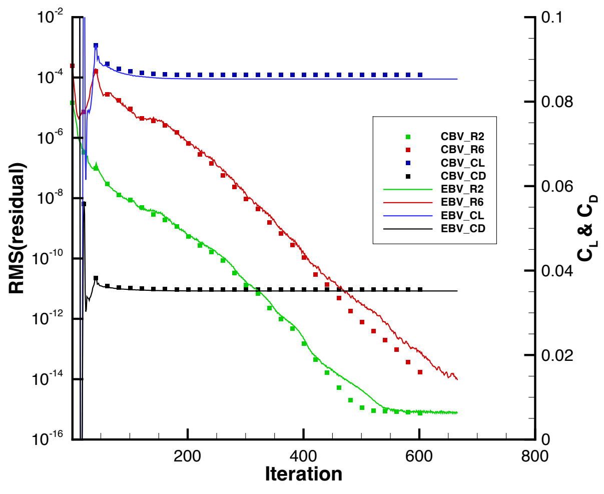
Table 8. Memory usage by baseline solver on prismatic-hexahedral grids.

Grid	CBV (kb)	EBV (kb)	EBV saving
PH1	160,048,308	153,040,636	4.4%
PH2	18,884,496	18,187,752	3.7%
PH3	2,372,840	2,277,576	4.0%

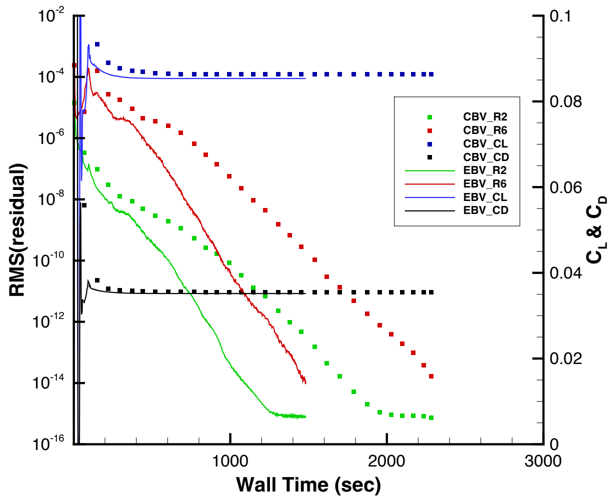
HANIM iterations

The strong nonlinear iterative solver, HANIM [31], presents unique challenges and opportunities for the EBV method. The HANIM's intention to operate at as high a CFL number as possible leads to an increased number of multicolor GS sweeps in the linear solver, possibly multiple search directions for the GCR solver, and relatively frequent nonlinear iteration failures, which decreases the fraction of viscous-kernel computations and challenges the EBV speedup. However, each HANIM iteration updates the meanflow and SA-neg Jacobians, which increases the fraction of viscous-kernel computations and creates opportunities for the EBV speedup. An accurate prediction of the EBV speedup for an individual HANIM iteration is not possible because HANIM iterations are not identical. There are many run-time decisions that HANIM makes based on comparison of floating-point numbers. Thus, timing of each HANIM iteration is sensitive to small details of discretization, previous solution, and input parameters. The HAMIM-CBV and HANIM-EBV convergence plots are expected to be visibly different, unlike the baseline CBV and EBV iteration plots shown in Figs. 3a and 4a. HANIM simulations have been conducted for the CBV and EBV methods on the PH2 grid. The maximum number of multicolor GS sweeps is set to 200 for the meanflow and SA-neg preconditioners. The preconditioner residual reduction target is set to 0.5. The GCR residual reduction target is set to 0.92. Only one GCR search direction is allowed.

Figure 5 illustrates convergence of the HANIM-EBV and HANIM-CBV iterations. Figures 5a and 5b show the convergence history of the residuals and aerodynamic coefficients versus HANIM iterations and wall time, respectively. The aerodynamic coefficients computed by HANIM converge to the same values as the coefficients computed with the baseline solver. Figure 5c shows the CFL history. As expected, the HANIM-CBV and HANIM-EBV residual convergence histories shown in Fig. 5a are visibly different; convergence histories of aerodynamic coefficients are more similar. Both HANIM-EBV and HANIM-CBV iterations meet the residual convergence target, achieve a high CFL number, and significantly reduce the time to solution in comparison to the baseline iterations (cf. Fig. 3b). The CFL range of the HANIM-CBV and HANIM-EBV solvers is about the same.

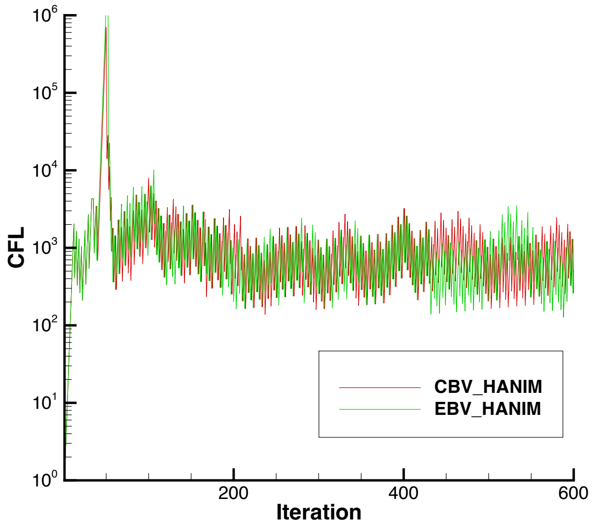


a. HANIM iterations



b. Wall time

Figure 5. HANIM iterations on PH2 grid.



c. CFL number

Figure 5. Concluded.

The statistics of the baseline and HANIM iterations on the PH2 grid is shown in Table 9. The HANIM convergence time and iteration count are compared with the time and iteration count of the baseline solver. HANIM significantly accelerates the solution process converging at least 38% faster than the baseline solver. The fraction of viscous-kernel computations in HANIM-CBV iterations decreases relative to the baseline-iteration fraction from 15.1% to 11.3%; the fraction of viscous-kernel computations in HANIM-EBV iterations is more similar to the baseline-iteration fraction, 4.2% vs. 3.9% (cf. Table 7). The HANIM-EBV solver converges 36.1% faster than the HANIM-CBV solver.

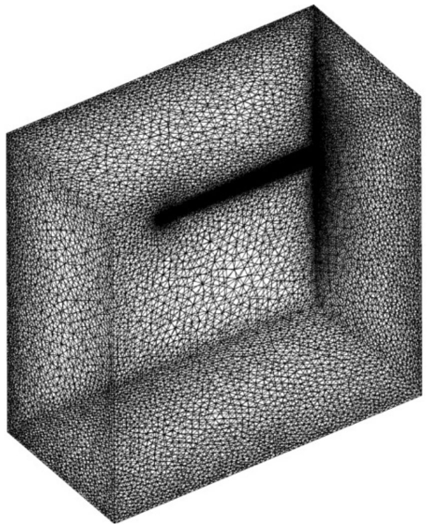
Table 9. Nonlinear iterations on PH2 grid.

	Baseline iterations		HANIM iterations		HANIM speedup
	Iterations	Wall time (sec)	Iterations	Wall time (sec.)	
CBV	4,005	3.782	618	2.311	39%
EBV	4,026	2.372	666	1.478	38%
EBV speedup		37.3%		36.1%	

VI. Juncture-Flow Model

References [19, 20] reported a hybrid EBV/CBV solution on a mixed-element grid for a flow around a NASA juncture-flow configuration. The EBV method was used on tetrahedra and the CBV method was used on other elements. No speedup was observed with the hybrid implementation because some of grid partitions contained only non-tetrahedral cells. In this section, we consider a solution corresponding to the new mixed-element implementation of the EBV method for the same flow on the same grid and compare it with the CBV solution. Recall that the SA-neg model used in this demonstration cannot capture the important characteristics of separation typical for juncture flows. The goal of the solutions reported here is not to analyze the actual flow phenomena, but rather to verify the EBV implementation and assess corresponding efficiency gains on a practical mixed-element grid.

The flow conditions for this test are the following: the freestream Mach number is 0.189, the Reynolds number is 2,400,000 based on the crank chord of 557.17 mm, the reference temperature is 288.84 K, and the angle of attack is 5°. A family of grids have been generated by Pointwise® using the Glyph script package GeomToMesh [41]. The specific mixed-element grid chosen for the study is Grid I [31, 42] that has 13,036,210 grid points, 16,645,072 tetrahedra, 20,368,758 prisms, and 112,989 pyramids. The volume and surface mesh are shown in Fig. 6.


a. Volume mesh

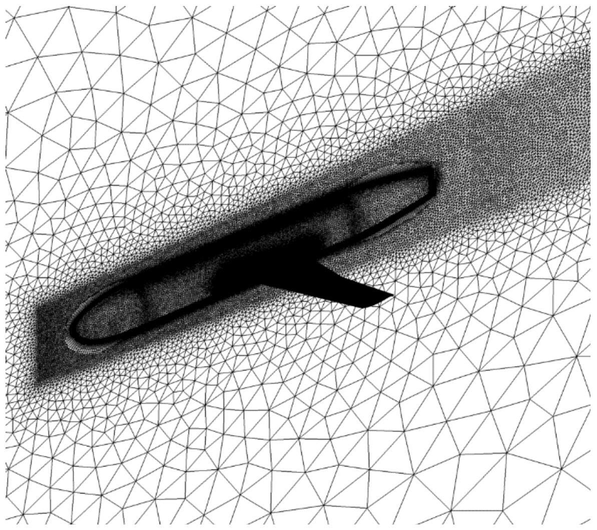

b. Surface mesh

Figure 6 Volume and surface mesh for Grid I of juncture flow model.

For the baseline iterations, the meanflow CFL of 50 and the SA-neg CFL of 30 are set to ensure convergence of all residuals to the level of 10^{-09}. The preconditioner performs 30 multicolor GS sweeps for the meanflow and SA-neg equations. For HANIM iterations, the following parameters are set. The maximum number of multicolor GS sweeps is 100 for the meanflow and SA-neg preconditioner equations. The preconditioner residual reduction target is 0.5, and the GCR residual reduction target is 0.92. Only one GCR search direction is allowed. The mixed-element solutions have been computed using 10 Intel Gold 6148 Skylake compute nodes (400 cores).

Table 10 shows aerodynamic coefficient computed from the converged CBV and EBV solutions. The matching digits of the EBV and CBV aerodynamic coefficients are highlighted by the bold font. The difference between the EBV and CBV lift coefficients is 0.17% and the difference between the EBV and CBV drag coefficients is 0.42%.

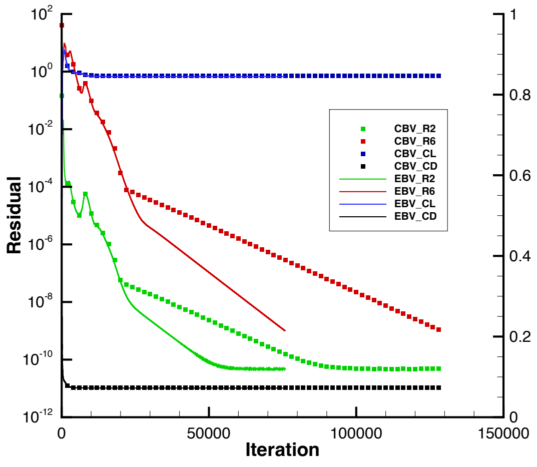
Table 11 provides a quantitative data about the nonlinear iterations and time required to reduce the rms norm of the residual below 10^{-9}. A great EBV speedup of more than 55% (a speedup factor of 2.3) is observed for the baseline and HANIM solutions. This speedup is higher than the speedup observed on prismatic-hexahedral grids considered in Section V. This is unexpected but can be explained by the observation that, on this mixed-element grid, the EBV solvers exhibit faster asymptotic convergence rate per nonlinear iteration than the CBV solvers, see Figs. 7 and 8 for the baseline solver and HANIM, respectively. Specifically, the baseline EBV solver requires 75,844 nonlinear iterations to reduce the rms norm of the residual below 10^{-9}; the baseline CBV solver requires 128,784 nonlinear iterations to reach this level of convergence. Some differences in iterative convergence between EBV and CBV solvers are expected, but such a significant disparity is probably not a general phenomenon. As expected, HANIM significantly accelerates convergence of both EBV and CBV solutions, providing speedup of more than 71% (a speedup factor of 3.4) over the baseline solver. Both HANIM-EBV and HANIM-CBV operate at high CFL numbers in the range of thousands, much higher than CFL chosen for the baseline solver.

Table 10. Aerodynamic coefficients for JFM.

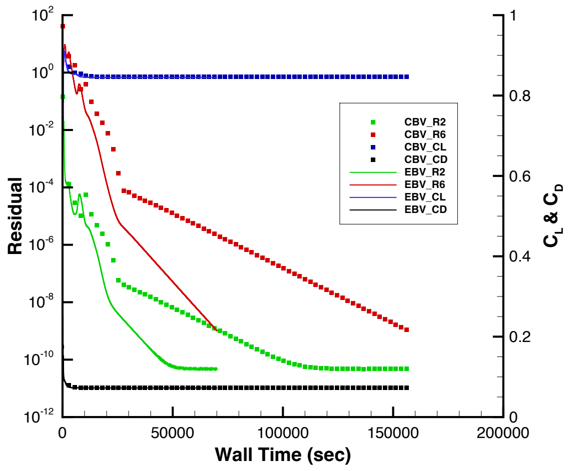
Lift Coefficient		**Drag Coefficient**	
CBV	**EBV**	**CBV**	**EBV**
0.846346	**0.84**48907	**0.072**7216	**0.072**3803

Table 11. Nonlinear iterations for JFM.

	Baseline iterations		**HANIM iterations**		**HANIM speedup**
	Iterations	**Wall time (sec)**	**Iterations**	**Wall time (sec.)**	
CBV	128,784	156,960	12,708	44,758	71%
EBV	75,844	70,080	7,153	18,104	74%
EBV speedup		55%		60%	

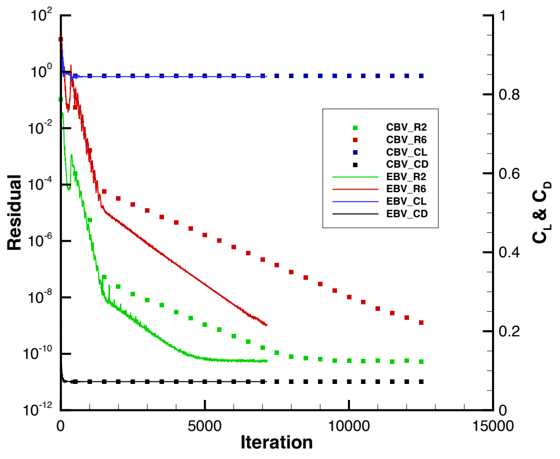


a. Nonlinear iterations

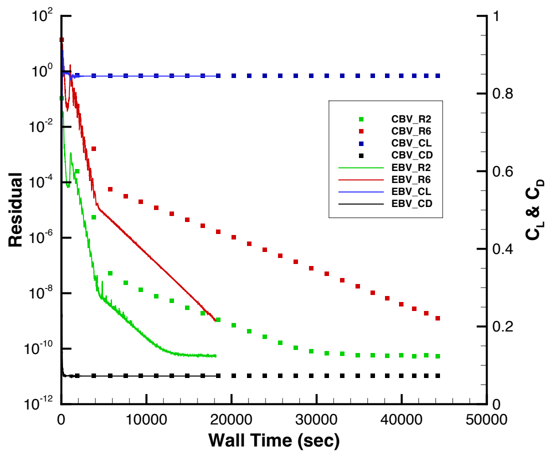


b. Wall time

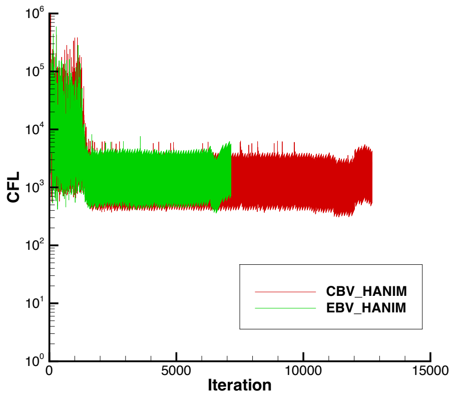
Figure 7. Baseline iterations on JFM mixed-element grid.



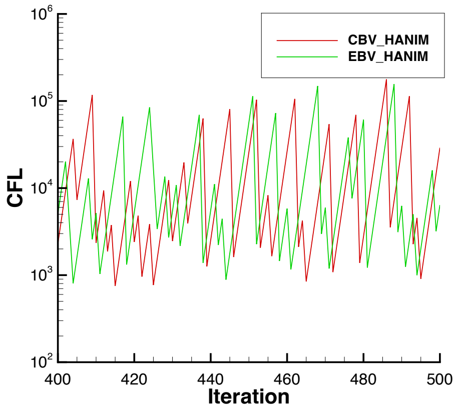
a. HANIM iterations



b. Wall time



c. CFL number (global view)



d. CFL number (zoomed view)

Figure 8. HANIM iterations on JFM mixed-element grid.

VII. High Enthalpy Hypersonic Flow Around a Hemisphere Cylinder

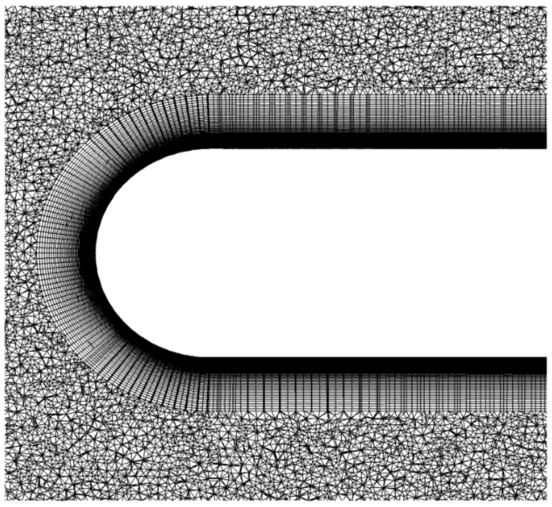
The flow considered in this section is a high-enthalpy flow around a blunt-body hemisphere-cylinder configuration corresponding to the freestream temperature $T_\infty = 450\ K$, pressure $p_\infty = 230$ Pa, Mach number $M_\infty = 9.8$, Reynolds number $Re_D \approx 14{,}000$ based on the cylinder diameter, and the angle of attack $\alpha = 0°$. A high-enthalpy high-speed flow is a compressible, chemically reacting flow that is governed by equations of conservation of the species, mixture momentum, and the total energy. The flow has been previously analyzed experimentally and computationally [43, 44]. Pressure is modeled as an ideal gas, heat transfer is modeled with Fourier's law, species diffusion is modeled using Fick's law, and the shear-stress is modeled with a Newtonian model. NASA polynomials [45] are used to compute thermodynamic properties on a per-species basis. The transport properties (diffusivity, viscosity, and thermal conductivity) are computed using collision integrals [46]. The full details of the equations, including the two-temperature model [47] that is used in this study can be found in Ref. [46]. Air is modeled using five species (N_2, O_2, NO, N, O) and five reactions [48]. The freestream mass fractions are set to standard air conditions. A laminar flow is assumed. The wall is modeled as a non-catalytic wall with a constant temperature of $T = 555.5\ K$ $(1000\ °R)$. Farfield is imposed on the inflow plane and extrapolation is imposed on the outflow plane.

In these computations, inviscid fluxes are discretized with the HLLE++ Riemann solver [44, 49] and use the van Albada flux limiter [50]. The discrete equations are integrated in time implicitly using a local time stepping with a maximum CFL number of 5.0. Chemistry, momentum, and energy equations are solved fully coupled, resulting in a block-sparse Jacobian matrix with 10×10 blocks.

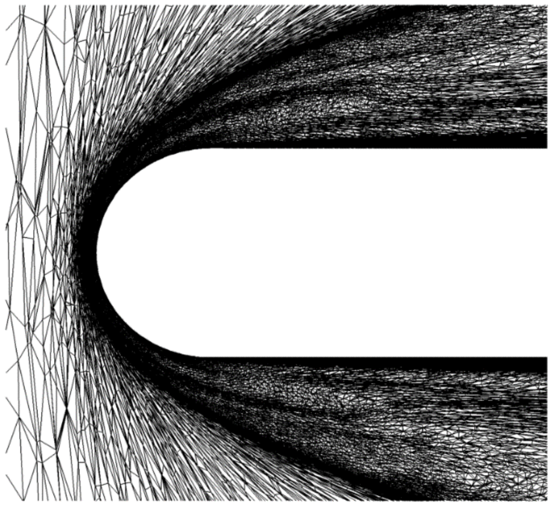
Two unstructured grids are considered. A general mixed-element grid has been generated by experts using the best practice and is denoted as the fixed grid. The finest adapted grid generated in a grid adaptation process governed by NASA's software refine [51] (see Ref. [44]) is denoted as the adapted grid. The grid statistics are shown in Table 12. The adapted grid is dominated by tetrahedra, the fixed grid has a comparable number of prisms and tetrahedra. For illustration, Fig. 9 shows the grid cross-section corresponding to $z = 0$ near the boundary layer. More details about the flow conditions and the grids can be found in Ref. [44].

Table 12. Grids for high-enthalpy flow around hemisphere-cylinder configuration.

Grid	Points	Tetrahedra	Prisms	CPU cores
Fixed	3,459,137	6,639,496	4,627,800	120
Adapted	4,567,239	23,562,449	1,028,320	160



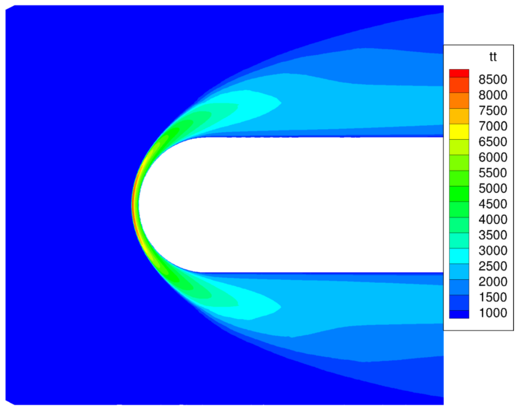
a. Fixed grid



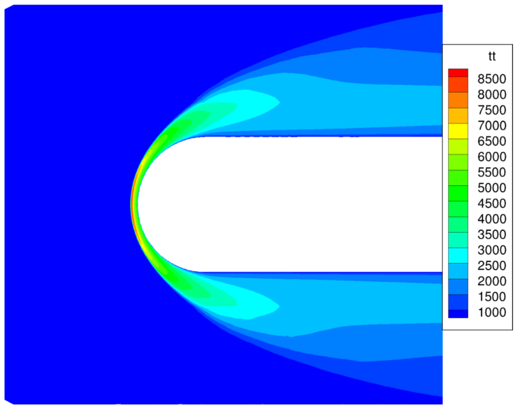
b. Adapted grid

Figure 9. Grid cross-section corresponding to $z = 0$.

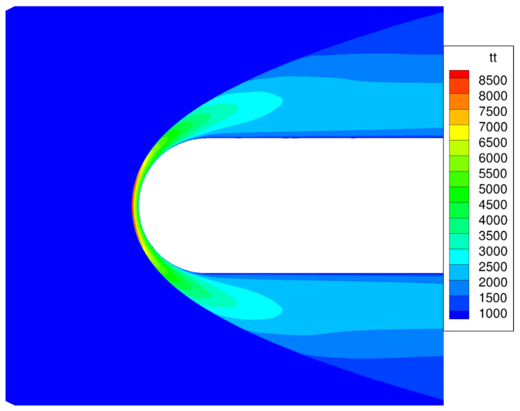
The EBV and CBV baseline solvers perform 4000 nonlinear iterations on the fixed and adapted grids. To illustrate similarities of the EBV and CBV solutions, Figs. 10 and 11 show the transitional-rotational temperature (tt) contours in the $z = 0$ cross-section and the hemisphere surface heating contours, respectively. The corresponding solutions are obtained after completion of 4000 iterations. While there are visible differences between solutions computed on the fixed and adapted grids, the EBV and CBV solutions computed on the same grids are almost identical.



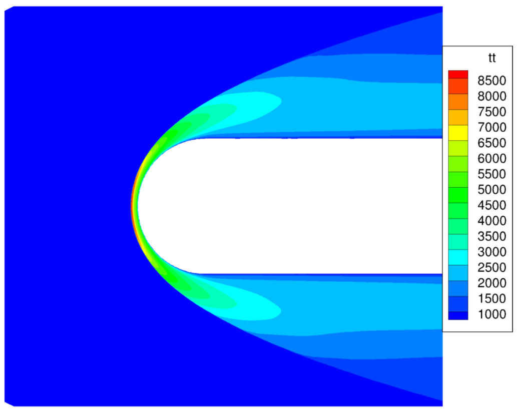
a. CBV, fixed grid



b. EBV, fixed grid

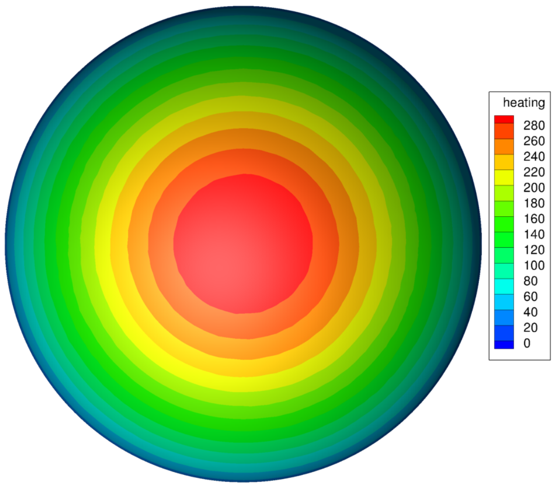


c. CBV, adapted grid

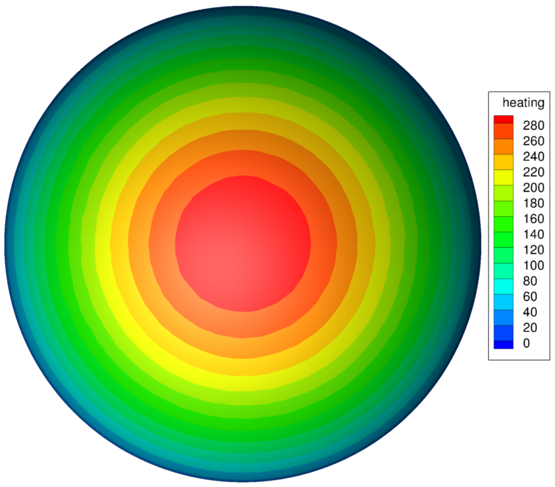


d. EBV, adapted grid

Figure 10. Transitional-rotational temperature (K) in cross-section corresponding to $z = 0$.

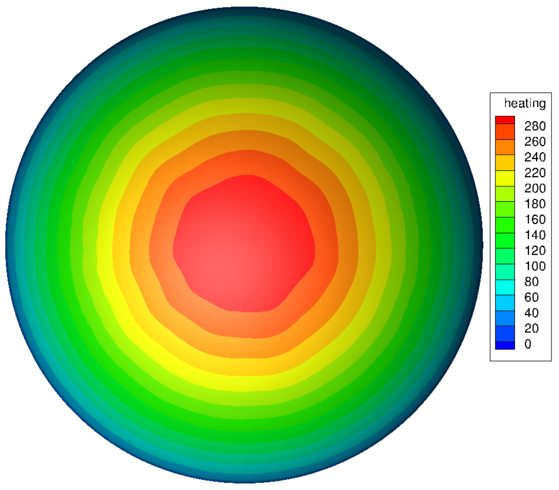


a. CBV, fixed grid

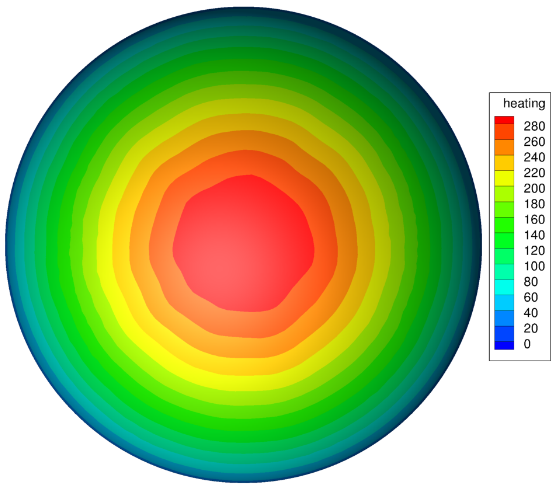


b. EBV, fixed grid

Figure 11. Hemisphere surface heating (W/cm^2).



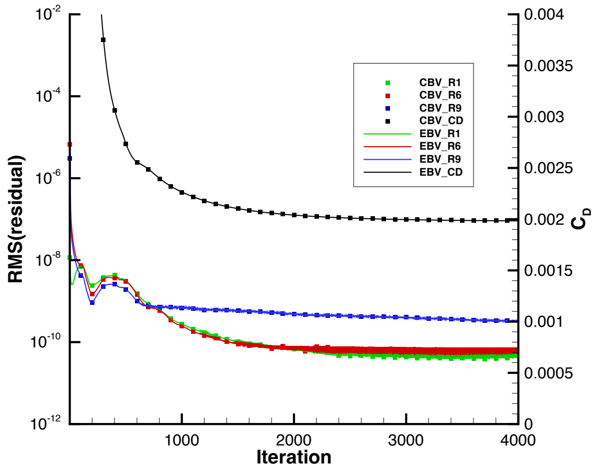
c. CBV, adapted grid



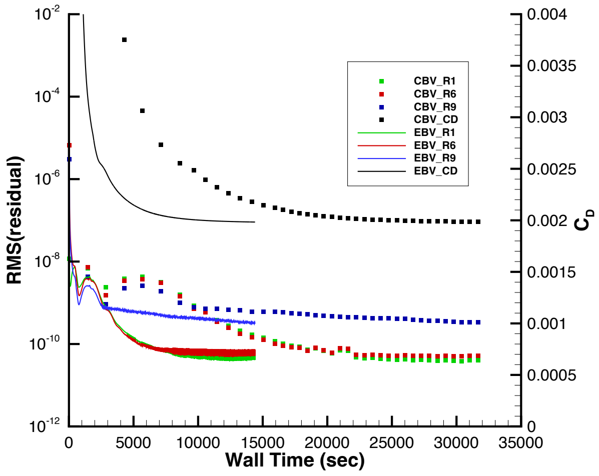
d. EBV, adapted grid

Figure 11. Concluded.

Figures 12 and 13 compare the iterative convergence on the fixed and adapted grids, respectively. The subscripts $R1$, $R6$, and $R9$ denote rms residuals of the N_2 mass, x-momentum, and total-energy conservation equations, respectively. The subscript CD and the axis label C_D denote the drag coefficient. In Figs. 12a and 13a, the residuals and the drag coefficient are plotted versus nonlinear iterations. The CBV and EBV residuals and drag coefficients are indistinguishable within the plotting accuracy. In Figs. 12b and 13b, the same data are plotted versus wall time. The EBV solutions dramatically outperform the corresponding CBV solutions on both the fixed and adapted grids.

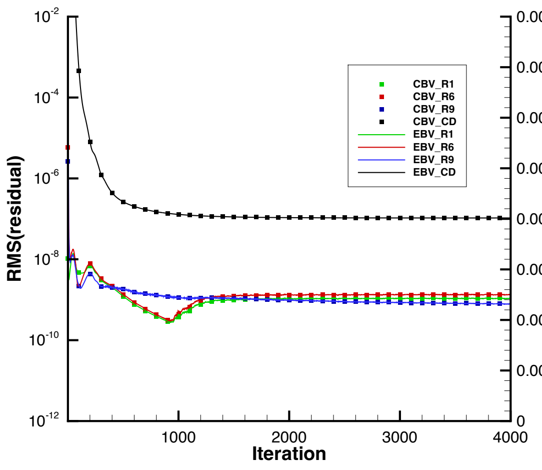


a. Nonlinear iterations

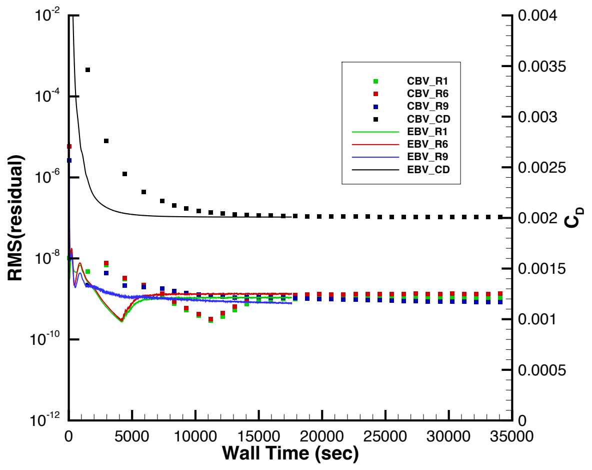


b. Wall time

Figure 12. Iterative convergence on fixed grid.



a. Nonlinear iterations



b. Wall time

Figure 13. Iterative convergence on adapted grid.

Table 13 shows aerodynamic coefficients computed after 4,000 iterations. The digits matching between the EBV and CBV coefficients computed on the same grid are highlighted by the bold font. The difference between the drag coefficients computed on the fixed and adapted grids is little more than 1%, the difference between the EBV and CBV drag coefficients computed on the same grid is 0.05% or less. The zero lift is expected for a flow at the zero angle of attack to an axisymmetric body. However, lack of grid symmetry results in the computed lift coefficients, which are very small, but not exactly zero.

Table 13. Aerodynamic coefficients for high-enthalpy flow around hemisphere-cylinder configuration.

Viscous method	Lift Coefficient		Drag Coefficient	
	Fixed grid	Adapted grid	Fixed grid	Adapted grid
CBV	**0.0000000**50	**-0.00000000**3	**0.00198**454	**0.0020**1037
EBV	**0.0000000**44	-**0.00000000**1	**0.00198**340	**0.0020**0876

The overall performance and memory usage of the baseline solver using the CBV and EBV methods are quantitatively assessed in Tables 14, 15 and 16. Table 14 shows the fraction of time spent in each type of viscous-kernel and linear-solver computations. In this chemically reacting flow, construction of the viscous Jacobian terms dominates the CBV solver. The viscous-term Jacobian computations consume more than half of the computing time on both fixed and adapted grids. The EBV method dramatically accelerates all viscous-kernel computations. The EBV speedup for the viscous-term Jacobian is more than 95% (a speedup factor of more than 20), reducing the viscous-kernel fraction from more than 56% to less than 7%. Although not shown, the EBV speedup for the linear solver is more than 21% on the fixed grid and 6.4% on the adapted grid. The dependence of the EBV linear-solver speedup on the grid composition is expected, the EBV method produces no speedup for the linear solver on tetrahedral cells. On both the grids, the EBV speedup for the linear solver is dwarfed by the EBV viscous-kernel speedup. As a result, the fraction of the linear solver grows significantly in the EBV solutions comparing with the linear-solver fraction in the CBV solutions.

Table 14 Fractions of viscous-kernel and linear-solver computations for high-enthalpy flow around hemisphere-cylinder configuration.

Grids	Viscous method	Iterations	Viscous flux fraction	Jacobian		Linear solver fraction
				Updates	Fraction	
Fixed	CBV	4,000	2.2%	1,668	54.0%	14.7%
	EBV	4,000	1.2%	1,671	5.7%	26.0%
Adapted	CBV	4,000	1.5%	2,246	58.6%	9.4%
	EBV	4,000	1.0%	2,238	6.0%	20.3%

Table 15 presents the time for completion of 4000 nonlinear iterations and the overall EBV speedup on the fixed and adapted grids. The EBV speedup exceeds 55.5% (a speedup factor of 2.25) on both the grids. Table 16 shows the memory usage. The EBV solver requires about 5% less memory than the CBV solver on the fixed grid and about 5% more memory than the CBV solver on the adapted grid. It is encouraging that even on grids that are heavily dominated by tetrahedra, the EBV memory increase is only 5%.

Table 15. Time and speedup for baseline solver for high-enthalpy flow around hemisphere-cylinder configuration.

Grid	CBV (sec)	EBV (sec)	EBV speedup
Fixed	32,269	14,350	55.5%
Adapted	40,721	17,686	56.6%

Table 16. Memory usage for baseline solver for high-enthalpy flow around hemisphere-cylinder configuration.

Grid	CBV (kb)	EBV (kb)	EBV saving
Fixed	78,216,700	74,577,776	4.7%
Adapted	96,410,656	101,339,772	-5.1%

VIII. Summary and Future Work

An efficient edge-based viscous (EBV) method for viscous-kernel computations has been extended to mixed-element grids. The viscous kernel includes evaluations of meanflow viscous fluxes, the diffusion terms in turbulence and chemistry models, and the corresponding Jacobian terms. The EBV method conducts viscous-kernel computations in an efficient loop over edges. The EBV method has been implemented in FUN3D, a practical, node-centered, unstructured-grid, finite-volume flow solver developed and supported at NASA. An initial implementation on tetrahedral grids was previously reported and demonstrated a multifold acceleration of viscous-kernel computations. The EBV method for mixed-element grids introduces virtual edges that connect the vertices of non-tetrahedral cells that are not connected by primal edges of the mixed-element grid. The virtual edges are introduced through a consistent division of non-tetrahedral cells into tetrahedra resulting in a derived tetrahedral grid. The derived tetrahedral grid has the same grid points and contains all primal edges of the mixed-element grid. The inviscid fluxes are evaluated in a loop over the primal edges of the original mixed-element grid, and the viscous fluxes are evaluated in a loop over the primal and virtual edges of the derived tetrahedral grid.

The EBV coefficients are defined for primal and virtual edges. Memory for six EBV coefficients is allocated for each interior edge of the grid, and memory for nine EBV coefficients is allocated for each boundary edge, although most of the boundary edges use only six EBV coefficients. This additional memory required for the EBV coefficients represents a fraction of the memory used by the baseline solver. The actual EBV memory footprint is often lower than the memory footprint of the baseline solver that uses a cell-based viscous (CBV) method on the same mixed-element grid. On mixed-element grids, the EBV residual accesses solutions at fewer grid nodes and reduces the number of off-diagonal terms in the Jacobian.

The EBV method has been verified and assessed by comparing EBV solutions with the solutions computed by the baseline CBV method for three benchmark flows, namely, a subsonic separated flow at a high angle of attack around a hemisphere-cylinder configuration, a subsonic flow around NASA juncture-flow model, and a high-enthalpy, chemically reacting, hypersonic flow around a hemisphere-cylinder configuration.

The EBV and CBV Reynolds-averaged Navier-Stokes (RANS) solutions for a subsonic separated flow around a hemisphere-cylinder configuration have been compared on a family of prismatic-hexahedral grids. The negative variant of the Spalart-Allmaras (SA-neg) turbulence model has been used. The grid convergence of the EBV solutions compares well with the grid convergence of the reference solutions reported at NASA's Turbulence Modeling Resource website. The residual reduction per iteration and converged solutions computed with the EBV and CBV methods are similar on all grids. The EBV solver has been profiled in detail, including the timing and speedup assessment for various components of the viscous kernel and the linear solver. The EBV speedup for the meanflow viscous fluxes and the corresponding Jacobian is more than 70% (a speedup factor of 3.3); the EBV speedup for SA-neg diffusion and Jacobian is at least 90% (a speedup factor of 10); the EBV speedup for the meanflow linear solver and the SA-neg linear solver is over 39% (a speedup factor of 1.6) and over 26% (speedup factor of 1.35), respectively. The EBV speedup for an individual nonlinear iteration ranges from 35% to 47%, the speedup is higher for iterations that update Jacobian. Overall, the EBV method provides between 37% and 41% reduction of time to solution for the baseline solver on the prismatic-hexahedral grids. The EBV method also results in an up to 9% reduction in the memory usage.

A strong hierarchical adaptive nonlinear iteration method (HANIM) has recently been developed to improve robustness of finite-volume solutions and accelerate convergence. The HANIM-EBV solver has provided an additional 36% reduction of time to solution on a prismatic-hexahedral grid in comparison with a highly efficient HANIM-CBV solver on the same grid.

The EBV method has been assessed in application to a benchmark turbulent flow around NASA's juncture-flow model (JFM) computed on a mixed-element grid. The aerodynamic coefficients computed from the converged EBV and CBV solutions differ by less than 0.47%. Somewhat surprisingly, the EBV solutions require significantly fewer nonlinear iterations to reduce root-mean-square norm of the residuals below the target level of 10^{-9}. The improved asymptotic convergence rate is observed for both the baseline solver and HANIM. Overall, the EBV speedup is 55% (a speedup factor of 2.3) for the baseline solver and 60% (a speedup factor of 2.5) for HANIM.

The EBV method has also been assessed for a laminar, high-enthalpy, chemically reacting, hypersonic flow over a blunt-body configuration. The assessment includes EBV and CBV solutions on two mixed-element grids: a fixed grid with 3.5 million grid points was generated by experts following the best practices and an adapted grid with 4.6 million grid points was generated by an automated grid adaptation process. The fixed grid has about 60% of tetrahedra and 40% of prisms; the adapted grid is dominated (more than 96% of all cells) by tetrahedral cells. The EBV and CBV solvers perform 4000 nonlinear iterations on each grid. The EBV and CBV solutions and the residual convergence versus iterations are indistinguishable within the plotting accuracy. The drag coefficients computed by the EBV and CBV solvers on the same grid differ by less than 0.05%. The EBV solver speedup is 55.7% (a speedup factor of 2.26) on the fixed grid and 57.5% (a speedup factor of 2.35) on the adapted grid. The EBV solver uses 5% less memory resources on the fixed grid and 5% more memory resources on the adapted grids. It is evident that on grids with a large fraction of non-tetrahedral cells, the Jacobian memory reduction enabled by the EBV method outpaces the EBV memory allocation for virtual edges and EBV coefficients.

There are several directions for future extensions of the EBV method. The modern computer architectures, e.g., graphics processing units (GPU), favor computations with a smaller memory footprint. Significant benefits are expected from porting mixed-element EBV solvers on GPU-based computers. Modern turbulence models introduce a quadratic constitutive relation (QCR) which is often implemented as a nonlinear correction to the shear stresses. The QCR correction involves a product of velocity gradient terms. Extension of the EBV method to viscosity terms that involve nonlinear functions of solution gradients is an important topic for future research. A solid theoretical foundation for accuracy and stability of the EBV solutions for various sets of flow equations is critical for establishing confidence in EBV solutions and is another active research direction. Practical mixed-element grids often include non-tetrahedral cells that are highly stretched, skewed, and twisted, resulting in highly non-planar quadrilateral faces. Such geometrical features complicate division of non-tetrahedral cells into tetrahedra and generation of corresponding virtual edges for the EBV method. The EBV virtual edges rely on a derived tetrahedral grid that is expected to have the same set of grid points and preserve all edges of the baseline mixed-element grid. Robust generation of valid tetrahedral grids from given mixed-element grids is another important topic of the future research.

Acknowledgments

The Transformative Tools and Technologies (TTT) project of the Transformative Aeronautics Concepts Program and the NASA Revolutionary Vertical Lift Technology (RVLT) project within the NASA Aeronautics Research Mission Directorate partially funded the work reported here. The three first authors were supported by the NASA Langley Research Center under cooperative agreement 80LARC17C0004 with the National Institute of Aerospace. The third author gratefully acknowledges support from the U.S. Army Research Office under the contract/grant number W911NF-19-1-0429.

References

[1] Barth T. J., "Numerical Aspects of Computing Viscous High Reynolds Number Flows on Unstructured Meshes," AIAA 91–0721, 1991. https://doi.org/10.2514/6.1991-721

[2] Luo H., Baum J. D., Lohner R., and Cabello J., "Adaptive Edge-Based Finite Element Scheme for the Euler and Navier-Stokes Equations on Unstructured Grids," AIAA 93-0336, 1993. https://doi.org/10.2514/6.1993-336

[3] Anderson W. K. and Bonhaus D. L., "An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids," *Computers and Fluids*, Vol. 23, No. 1, 1994, pp. 1-21. https://doi.org/10.1016/0045-7930(94)90023-X

[4] Eliasson P., “EDGE, a Navier-Stokes Solver for Unstructured Grids,” FOI-Swedish Defense Research Agency, FOI-R-0298-SE, December 2001. https://www.foi.se/rest-api/report/FOI-R--0298--SE
[5] Mavriplis D.J., “Grid Resolution Study of a Drag Prediction Workshop Configuration Using the NSU3D Unstructured Mesh Solver,” AIAA 2005-4729, 2005. https://doi.org/10.2514/6.2005-4729
[6] Schwamborn D., Gerhold T., and Heinrich R. K., “The DLR TAU Code: Recent Applications in Research and Industry,” Proceedings of ECCOMAS CFD 2006, Delft, Netherlands, 2006, Paper 619. https://elib.dlr.de/22421/
[7] Braaten M. E. and Connel S. D., “Three-Dimensional Unstructured Adaptive Multigrid Scheme for the Navier-Stokes Equations,” *AIAA Journal*, Vol. 34, No. 2, February 1996, pp. 281–290. https://doi.org/10.2514/3.13062
[8] Haselbacher A. and Blazek J., “Accurate and Efficient Discretization of Navier-Stokes Equations on Mixed Grids,” *AIAA Journal*, Vol. 38, No. 11, 2000, pp. 2094–2102. https://doi.org/10.2514/2.871
[9] Nishikawa H., “Beyond Interface Gradient: A General Principle for Constructing Diffusion Schemes," AIAA Paper 2010-5093, 2010. https://doi.org/10.2514/6.2010-5093
[10] Diskin, B., Nishikawa, H., and Liu, Y., “Analysis of Edge-Based Method for Diffusion,” 11th International Conference on Computational Fluid Dynamics, Maui, HI, 11-15 July 2022. (submitted).
[11] Biedron R. T., Carlson J. R., Derlaga J. M., Gnoffo P. A., Hammond D. P., Jones W. T., Kleb B., Lee-Rausch E. M., Nielsen E. J., Park M. A., Rumsey C. L., Thomas J. L., Thompson K. B., Walden, A. C., Wang, L., and Wood W. A., “FUN3D Manual: 13.7,” NASA Technical Memorandum 2020-5010139, 2020. https://fun3d.larc.nasa.gov/chapter-2.html#manual_13_7
[12] Diskin B., Ahmad N. N., Anderson W. K., Derlaga J. M., Pandya M. J., Rumsey C. L., Wang L., Wood S. L., Liu Y., Nishikawa H., and Galbraith M.C., “Verification Test Suite for Spalart-Allmaras QCR2000 Turbulence Model,” AIAA 2021-1552, 2021. https://doi.org/10.2514/6.2021-1552
[13] Diskin B., Anderson W. K., Pandya M. J., Rumsey C. L., Thomas J.L., Liu Y., and Nishikawa H., “Grid Convergence for Three Dimensional Benchmark Turbulent Flows,” AIAA 2018-1102, 2018. https://doi.org/10.2514/6.2018-1102
[14] Rumsey C. L., Carlson J.-R., and Ahmad N. N., “FUN3D Juncture Flow Computations Compared with Experimental Data,” AIAA 2019-0079, 2019. https://doi.org/10.2514/6.2019-0079
[15] Abdol-Hamid K.S., Carlson J.-R., Rumsey C. L., Lee-Rausch E. M., and Park M. A., “Sixth Drag Prediction Workshop Results Using FUN3D with k-kL-MEAH2015 Turbulence Model,” Journal of Aircraft, Vol. 55, No. 6, pp. 1458-1468, 2018. https://doi.org/10.2514/1.C034481
[16] Diskin B., Thomas J. L., Nielsen E. J., Nishikawa H., and White J. A., “Comparison of Node-Centered and Cell-Centered Unstructured Finite-Volume Discretizations: Viscous Fluxes,” AIAA Journal, Vol. 48, No. 7, pp. 1326-1338. 2010. https://doi.org/10.2514/1.44940
[17] Thomas J. L., Diskin B., and Ramsey C. L., “Towards Verification of Unstructured-Grid Solvers,” AIAA Journal, Vol. 46, No. 12, pp. 3070-3079, 2008. https://doi.org/10.2514/1.36655
[18] Diskin B. and Thomas J. L., “Accuracy Analysis for Mixed-Element Finite-Volume Discretization Schemes,” NIA Report 2007-08. https://fun3d.larc.nasa.gov/papers/nia_2007_08.pdf
[19] Liu Y., Diskin B., Anderson W. K., Nielsen, E. J., and Wang, L., “Edge-Based Viscous Method for Node-Centered Formulations,” AIAA 2021–2728, 2021. https://doi.org/10.2514/6.2021-2728
[20] Liu Y., Diskin B., Nishikawa H., Anderson, W. K., Nielsen, E. J., and Wang, L., “Edge-Based Viscous Method for Node-Centered Finite-Volume Formulations,” *AIAA Journal* (submitted 02/11/2022).
[21] Dompierre J., Labb P., Vallet M.-G., and Camarero R., “How to Subdivide Pyramids, Prisms and Hexahedra into Tetrahedra,” Ecole Polytechnique Montreal, Rapport CERCA R99-78, 1999. https://dblp.org/rec/conf/imr/DompierreLVC99.html
[22] Roe P. L., “Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes,” *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372. https://doi.org/10.1016/0021-9991(81)90128-5
[23] Burg C. O. E., “Higher Order Variable Extrapolation for Unstructured Finite Volume RANS Flow Solvers,” AIAA Paper 2005–4999, 2005. https://doi.org/10.2514/6.2005-4999
[24] van Leer B., “Towards the Ultimate Conservative Difference Scheme, V. A Second Order Sequel to Godunov’s Method,” *Journal of Computational Physics*, Vol. 32, No. 1, 1979, pp. 101–136. https://doi.org/10.1016/0021-9991(79)90145-1
[25] Allmaras S. R., Johnson F. T., and Spalart P. R., "Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model," ICCFD7-1902, 7th International Conference on Computational Fluid Dynamics, Big Island, Hawaii, 9-13 July 2012. https://www.iccfd.org/iccfd7/assets/pdf/papers/ICCFD7-1902_paper.pdf

[26] Spalart P. R. and Allmaras S. R., “A One-Equation Turbulence Model for Aerodynamic Flows,” *Recherche Aerospatiale*, No. 1, 1994, pp. 5-21. https://turbmodels.larc.nasa.gov/Papers/RechAerosp_1994_SpalartAllmaras.pdf
[27] Haselbacher, A. C., “A Grid-Transparent Numerical Method for Compressible Viscous Flow on Mixed Unstructured Meshes,” Ph.D. thesis, Loughborough University, December 1998. https://hdl.handle.net/2134/7257
[28] Thomas J. L., Diskin B., and Nishikawa H., “A Critical Study of Agglomerated Multigrid Methods for Diffusion on Highly Stretched Grids,” *Computers & Fluids,* Vol. 41, No. 1, 2011, pp. 82-93. https://doi.org/10.1016/j.compfluid.2010.09.023
[29] Carlson J.-R., “Inflow/Outflow Boundary Conditions with Application to FUN3D,” NASA/TM–2011-217181, 2011. https://fun3d.larc.nasa.gov/papers/NASA-TM-2011-217181.pdf
[30] van Leer B. “Flux-Vector Splitting for the Euler Equations,” ICASE Report 82-30, 1982.
[31] Wang L., Diskin B., Nielsen E. J., and Liu Y., “Improvements in Iterative Convergence of FUN3D Solutions,” AIAA 2021-0857, 2021. https://doi.org/10.2514/6.2021-0857
[32] Pandya M., Diskin B., Thomas J., and Frink N., “Assessment of USM3D Hierarchical Adaptive Nonlinear Method Preconditioners for Three-Dimensional Cases,” *AIAA Journal*, Vol. 55, No. 10, 2017, pp. 1–16. https://doi.org/10.2514/1.J055823
[33] Pandya M. J., Diskin B., Thomas J. L., and Frink N. T., “Improved Convergence and Robustness of USM3D Solutions on Mix-Element Grids,” *AIAA Journal*, Vol. 54, No. 9, 2016, pp. 2589–2596. https://doi.org/10.2514/1.J054545.
[34] Pandya M. J., Jespersen D. C., Diskin B., Thomas J. L., and Frink N. T., “Efficiency of Mixed-Element USM3D for Benchmark Three-Dimensional Flows,” *AIAA Journal*, Vol. 59, No. 8, 2021, published online June 21, 2021. https://doi.org/10.2514/1.J059720
[35] Pandya M. J., Jespersen D. C., Diskin B., Thomas J. L., and Frink N. T., “Verification and Scalability of Mixed-Element USM3D for Benchmark Three-Dimensional Flows,” *AIAA Journal*, published online August 30, 2021. https://doi.org/10.2514/1.J060064
[36] Knoll D. A. and Keyes D. E., “Jacobian-Free Newton–Krylov Methods: A Survey of Approaches and Applications,” *Journal of Computational Physics*, 193 (2004) 357–397. https://doi.org/10.1016/j.jcp.2003.08.010
[37] van der Vorst H. A. and Vuik C. “GMRESR: A Family of Nested GMRES Methods,” *Numerical Linear Algebra with Applications*, Vol. 1, 1994, pp. 369-386. https://doi.org/10.1002/nla.1680010404
[38] Lucas P., van Zuijlen A. H., and Bijl H., “Fast Unsteady Flow Computations with a Jacobian-free Newton-Krylov Algorithm," *Journal of Computational Physics*, Vol. 229, No. 24, 2010, pp. 9201-9215. https://doi.org/10.1016/j.jcp.2010.08.033
[39] Ceze M. and Fidkowski K., “A Robust Adaptive Solution Strategy for High-Order Implicit CFD Solvers," AIAA Paper 2011-3696, 2011. https://doi.org/10.2514/6.2011-3696
[40] Tsieh T., “An Investigation of Separated Flow about a Hemisphere Cylinder at 0- to 19-Deg Incidence in the Mach Number Range of 0.6 to 1.5,” AEDC-TR-76-112, 1976. https://apps.dtic.mil/sti/pdfs/ADA073451.pdf
[41] Karman S. and Wyman N., “Automatic Unstructured Mesh Generation with Geometry Attribution,” AIAA Paper 2019-1721, 2019. https://doi.org/10.2514/6.2019-1721
[42] Rumsey C. L., Lee H. C., and Pulliam T. H., “Reynolds-Averaged Navier-Stokes Computations of the NASA Juncture Flow Model Using FUN3D and OVERFLOW,” AIAA Paper 2020–1304, 2020. https://doi.org/10.2514/6.2020-1304
[43] Josyula E., and Shang J. S., “Computation of Nonequilibrium Hypersonic Flowfields around Hemisphere Cylinders,” *Journal of Thermophysics and Heat Transfer*, Vol. 7, No. 4, 1993, pp. 668–679. https://doi.org/10.2514/3.476
[44] Nastac G., Tramel R., and Nielsen E. J., “Improved Heat Transfer Prediction for High-Speed Flows over Blunt Bodies using Adaptive Mixed-Element Unstructured Grids” AIAA Paper 2022-0111, 2022. https://doi.org/10.2514/6.2022-0111
[45] McBride B. J., Gordon S., and Reno M. A., “Coefficients for Calculating Thermodynamic and Transport Properties of Individual Species,” NASA Technical Memorandum 4513, 1993. https://ntrs.nasa.gov/api/citations/19940013151/downloads/19940013151.pdf
[46] Gnoffo P. A., Gupta R. N., and Shinn J. L., Conservation Equations and Physical Models for Hypersonic Air Flows in Thermal and Chemical Nonequilibrium, NASA Technical Paper 2867, February 1989. https://ntrs.nasa.gov/api/citations/19890006744/downloads/19890006744.pdf

[47] Park C., “Assessment of Two-Temperature Kinetic Model for Ionizing Air,” Journal of Thermophysics and Heat Transfer, Vol. 3, No. 3, 1989, pp. 233–244. https://doi.org/10.2514/3.28771
[48] Park C., “On Convergence of Computation of Chemically Reacting Flows,” AIAA Paper 1985-0247, 1985. https://doi.org/10.2514/6.1985-247
[49] Tramel R., Nichols R., and Buning P., “Addition of Improved Shock-Capturing Schemes to OVERFLOW 2.1,” AIAA Paper 2009-3988, 2009. https://doi.org/10.2514/6.2009-3988
[50] van Albada G.D., van Leer B., and Roberts W.W., “A Comparative Study of Computational Methods in Cosmic Gas Dynamics”, *Astronomy and Astrophysics*, Vol. 108, No. 1, 1982, pp. 76–84. https://link.springer.com/chapter/10.1007/978-3-642-60543-7_6
[51] Park M. A., “Anisotropic Output-Based Adaptation with Tetrahedral Cut Cells for Compressible Flows,” Ph.D. thesis, Massachusetts Institute of Technology, Sep. 2008. https://dspace.mit.edu/handle/1721.1/46363